WHITE PAPER

Overcoming the Challenges of Testing in Secure Environments

KEYSIGHT

Table of Contents

* Cover image courtesy of the U.S. Navy; photo by Mass Communication Specialist 1st Class Jeff Troutman

Building and testing software in secure environments is uniquely challenging. Projects can be top secret, teams often work in silos, and quality standards are sky-high and unwavering. Simply put, there is no margin for error when lives and national security are at stake.

Various structural and technological factors present obstacles that can make using the most modern tools and methods difficult.

Advances in test automation have made it easier to hit quality goals and release software in shorter cycles. But the challenges of automated testing in secure environments can deter teams looking to modernize their workflows and maximize quality.

The good news is that you can overcome these challenges. This guide will show you how.

Test Automation: No Longer Just a Nice-to-Have

The past few years have seen a big push for modernizing development practices in the defense industry. The US Department of Defense (DoD) launched its Digital Modernization Strategy in July 2019, and a number of directives and initiatives to push software development into the 21st century have followed.

While these initiatives include various practices and technologies, test automation is particularly important. And it's no wonder, given that DevOps practitioners cite testing as the number one cause of delays in software releases. To drive the point home, let's take a look at DoD Instruction 5000.87, Operation of the Software Acquisition Pathway:

"Automated testing and operational monitoring should be used as much as practicable for user acceptance and to evaluate software suitability, interoperability, survivability, operational resilience, and operational effectiveness." [1]

This push for automation is already yielding positive results, with Nicolas Chaillan, Chief Software Officer of the US Air Force, reporting a 30% reduction in program time. [2]

If the benefits are great and the requirement so clear, why isn't test automation more widespread in the industry? Let's look at the obstacles.

Test Automation Challenges in Defense

To understand the slow adoption of test automation, you need to acknowledge the unique conditions in the defense industry:

- need for secrecy
- diversity of technologies and systems
- integration of multivendor software systems
- inability to install testing software on sensitive machines
- difficulty of automating multifactor authentication systems

---

1 A maturing DevSecOps landscape. (2021). Retrieved from Gitlab website: https://learn.gitlab.com/c/2021-devsecops-report?x=u5RjB_

2 DevSecOps security in the DoD | DevOps radio episode 72. (n.d.). Retrieved from https://www.cloudbees.com/resources/devops-radio/episode-72-devsecops-security

Secrecy

Much of the work in the defense industry is classified. That complicates the adoption of test automation and creates several issues:

- Testers sometimes don't have access to the underlying source code. A tester may access the system under test (SUT) via an interface that prevents humans or automation tools from inspecting the code.
- Teams often work in silos. There isn't much that different teams can say about their work without crossing this secrecy barrier.
- National security demands preclude development and testing practices common in the civilian world. While there is a big push toward DevSecOps across the DoD, some programs lag in adopting these practices. Consumer or business apps can silently receive updates overnight, but software used in the field rarely gets mid-deployment updates.

Variety of technologies involved

The tech stacks of defense systems can look very different from those found in the civilian world. Software may need to interact and integrate with systems or components designed decades ago. To achieve the cost savings associated with test automation, you need solutions that can reliably handle and automate these diverse technologies.

Where standard automation tools fail

Legacy systems are impossible to test with most test automation tools. One reason is that tools may require access to the source code or certain object layer properties or identifiers. This simply isn't possible when you're dealing with a command line interface, for instance. Selenium, which is the most popular tool for testing web applications, would be useless for automating such a system.

Integration of multivendor software systems

Software rarely, if ever, operates in isolation. Even seemingly simple systems may comprise dozens of parts. Yes, each separate part needs to work. But more importantly, the system as a whole needs to function properly and predictably. This necessitates end-to-end testing, which is often a highly manual process.

Test automation tools must test these end-to-end processes that touch a variety of software and hardware systems. These systems can vary greatly in terms of the programming language or platform they use. Therefore, to accommodate the diversity of technologies present in defense systems, test automation tools must be agnostic to operating system, programming language, and interface type.

Inability to install testing software on sensitive machines

Because of the secretive and sensitive nature of many defense systems, installing testing tools on the machines running the software in question isn't an option.

The answer is a two-system testing model that allows the testing solution to sit on one computer that securely connects to the SUT. Depending on the system requirements, technologies such as Remote Desktop Protocol (RDP) or Virtual Network Computing (VNC) can achieve this connection.

Difficulty of automating multifactor authentication systems

To further complicate matters, many defense systems require multifactor authentication (MFA) or two-factor authentication (2FA) for added security. This presents yet another issue for automation software.

The good news is that there are solutions that can securely access and automate systems that use MFA technology, such as smart cards or Common Access Cards (CACs). That means testers don't need to be physically close to the device in question, thus removing a major obstacle to testing secure systems.

User access problems in automation

In contrast to civilian systems, military software rarely has superusers, or admin accounts with wide-ranging permissions. This creates a need to log in to systems with a variety of user credentials to access the various components included in an end-to-end test. Therefore, the testing solution should be able to retrieve securely stored credentials from the system keychain in order to have proper access to the necessary systems.

The Power of Non-invasive Automation

You may be wondering how to overcome these obstacles. The answer is simple: non-invasive automation.

In short, non-invasive automation means that the solution can interact with the software and systems without jeopardizing any security requirements. This allows for robust automated end-to-end testing of highly secure systems.

A combination of two techniques achieves non-invasive automation:

- A two-system model allows the testing software to sit on a machine that's separate from the SUT.
- Testing is done via the user interface (UI) rather than through access to the source code or object layer, both of which can be obscured or unavailable to standard automation tools.

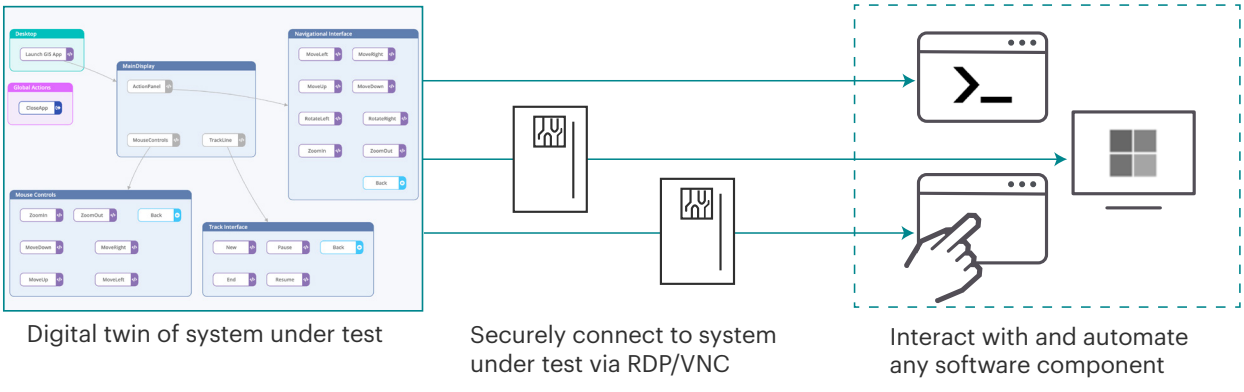

Figure 1. Diagram of two-system testing model. Eggplant (left) securely connects via RDP or VNC connection and/or smart card/CAC to software on various devices (right).

Testing at the UI level

What does it mean to test at the UI level? Simply put, it means that the automation software can interpret and intelligently interact with the UI of the SUT. This process works via a combination of optical character recognition (OCR) and computer vision. OCR allows the automation software to read the on-screen text. Computer vision scans the interface for visual elements, buttons, fields, values, colors, and more to make sense of what appears on screen. Automation tools that test at the UI level can also access any object properties available or exposed to hone their understanding of the UI.

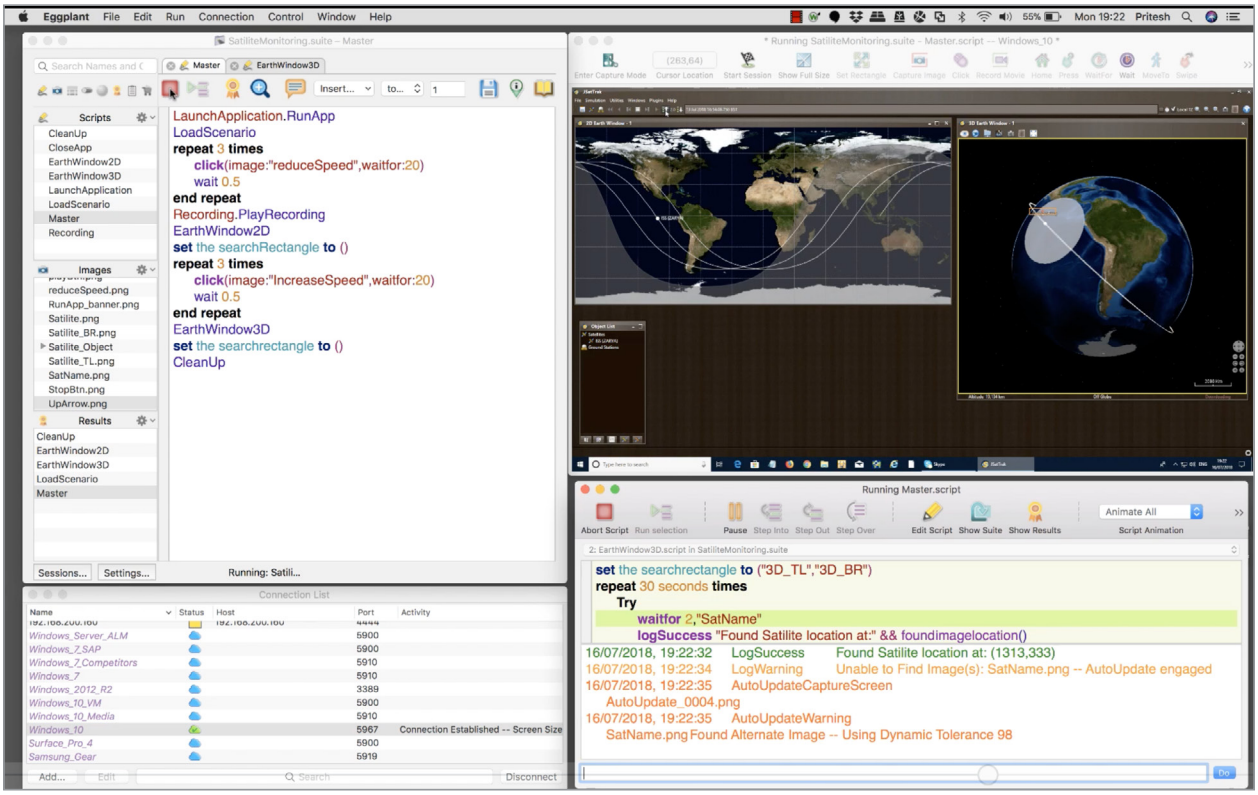

Figure 2. Eggplant test automation software (left and bottom right) drives actions in the GUI of a satellite monitoring application (top right)

Automation in practice

Test automation software isn't just for testing. Software teams have also had great success using these tools for processes such as system configuration and evaluation. Let's look at two examples.

Daily configuration

Testing labs at defense contractors are complex places. At one such contractor, one of its labs required a three-hour setup process every morning before any testing work could start. This process was a giant barrier to getting work done quickly, as it meant that the real testing work could only begin after lunch.

However, the team could automate this setup and configuration process through the creative use of a test automation solution. Instead of taking all morning, once initiated, the setup was ready to go within 15 minutes.

Evaluation

Another major stage of defense contracts is the formal handover process, which includes strict evaluation criteria. After all, if the government is paying millions of dollars for the delivery of a project, it needs to make sure the system functions as specified.

While it can vary by project, a typical evaluation process looks like this: An inspector, whether from the government or other contractual counterpart, travels to the development team's lab. The inspector will then sit with someone from the local team (perhaps a quality assurance, or QA, lead) and watch as a series of tests are conducted. These supervised demonstrations often last longer than a week and, while crucial to satisfying the contract, are tedious, to say the least.

However, using a test automation tool, QAs can prepare for these evaluations by recording all of the actions and workflows subject to testing. Then, when the inspector comes, they simply press play and together watch as the tests are completed. Because all the actions are automated and driven by a computer, testing is significantly faster, with no human error. Using this technique, one such contractor passed in only three days instead of the two weeks it previously took. And with the pandemic inhibiting travel, remote evaluations are possible.

The importance of exploratory testing: unearthing the unknown unknowns

Much of the testing effort and evaluation process focuses on linear, directed testing. This testing involves following a series of steps to demonstrate that certain functions and workflows function properly.

In practice, however, problems often arise in unexpected places. The testing surface — that is, the totality of possible test cases — can increase exponentially when you account for the complexity of applications and the systems they work in. It often falls on the plate of manual testers to haphazardly click around an application, trying to find and document anomalous behavior and bugs. Even worse, only predetermined paths may undergo testing, with potential issues lurking out of sight.

* Cover image courtesy of the U.S. Navy; photo by Mass Communication Specialist 1st Class Jeff Troutman

Automated exploratory testing at the UI level

Again, the workaround comes down to being able to test an application at the UI level. Test automation tools that can interpret and interact with the interface are able to proactively click around the SUT. This means testing out novel user journeys and potentially uncovering bugs that linear tests or manual testers would otherwise miss. Because test automation software automatically executes these tests, it's easy for QA teams to document and retrace the steps that led to bugs.

When you add the automation aspect to UI-level exploratory testing, you can quickly unlock millions of test cases and scenarios. In doing so, you can say goodbye to unknown unknowns in your software. Of course, in more complex settings, it will still be necessary to prioritize the testing of the most critical functional areas. However, it's only possible to prioritize test cases when you are actually aware of the much wider set of user journeys that you can test. This leads teams to a vastly better understanding of the true quality of the software they are delivering.

* Image courtesy of the U.S. Navy; photo by Clinton Beaird

Choosing a Test Automation Solution

It's not enough to work around the obstacles to test automation in secure development environments. You also want test automation to transform your ability to deliver quality software in shorter cycles. To achieve this, look out for the following when choosing a test automation solution:

Non-invasive testing

Non-invasive testing allows rigorous testing of highly secure systems without exposing any of the underlying code. By choosing a test automation tool that you can install on a separate machine and that lets you interact with the UI via a secure connection (RDP or VNC, depending on the use case), you can put your application through its paces without any breach of security.

2FA via smart cards or CACs

Because many systems in secure environments require 2FA via smart cards or CACs, your automation solution must accommodate these measures to securely access and test your applications. In addition, the automation software must handle logging in with different credentials to properly access each component when conducting end-to-end tests.

Device, operating system, and language agnostic

Some tools are meant to test Python or Java, and they do this very well. However, when you need to conduct secure end-to-end testing of systems written in various languages on platforms that may include Windows, Linux, iOS devices, or legacy systems such as command-line interfaces, you need a testing solution that is platform agnostic.

Broad and deep coverage of testing surface

The first project for QAs adopting a new test automation solution is to import and configure existing test cases. However, while these test cases may represent the most common user journeys, countless workflows remain untested. Therefore, it's important that your test automation solution allows you to conduct both deep and broad tests of applications and the systems they operate in. In addition to thoroughly covering your most essential user journeys, your testing solution should also be able to conduct these tests:

- all nodes — that is, any action, state, or variable value possible in the application
- all pairs, which covers pairs of actions taken in the application
- extended test runs of three or more nodes
- full exploratory tests that cover all possible user journeys through the application

By having a better grasp of the coverage of the application, you can better understand its overall quality. This allows a development team to integrate a testing learning loop into its workflow to continually hone and perfect the software in successive sprints.

Automation of demonstration and evaluation phase

Choosing the right test automation solution will allow you to streamline and drastically shorten the evaluation phase of the development project. Preconfiguring the tests that will be part of the evaluation will save time for both your team and whoever is conducting the evaluation. This means delivering software contracts in less time with fewer last-mile problems.

Eggplant

Eggplant, from Keysight Technologies, has been serving leading innovators in the defense and aerospace sectors for well over a decade.

Pioneer in non-invasive testing

Eggplant is the first test automation solution to master AI-driven graphical user interface (GUI) automation. NASA, the US Army, and numerous defense contractors have used Eggplant to test some of the most sensitive systems around. Eggplant's industry-leading approach combines OCR and computer vision to non-invasively test and visually validate the user experience. Eggplant can identify anomalous behavior and feed it back
to developers to create a virtuous feedback loop where quality continually improves.

Secure two-system approach

Eggplant is installed on a separate machine from the system under test. Then, via a secure connection, Eggplant can interpret and drive the SUT. In addition, because Eggplant can automate common 2FA technologies such as smart cards, CACs, and SMS codes, users can securely test and automate even the most sensitive systems.

Ease of use

A wide variety of people can use Eggplant. Traditional automation tools require robust programming skills. Eggplant, on the other hand, allows anyone to build a model of the SUT. This broadens the circle of those who can create testing models so that anyone can contribute to the quality of the project, not just developers and QA professionals.

Eggplant also includes analytics tools to easily give a snapshot overview of the progress of tests, coverage of your application, and performance against quality metrics.

Case Study: US Army selects Eggplant for test automation

Battlegrounds are constantly evolving, developing parallel to advances in technology. With computer systems central in today's frontline defense, modern warfighters rely on software to provide time-sensitive intelligence and targeting information to protect against adversaries.

For the US Army Communications-Electronics Command Software Engineering Center (SEC), Joint Tactical Terminals (JTT) delivers this critical information to its commanders in battle. JTT is a "family of software-programmable radios ... often integrated into larger, more complex systems."

In 2018, a critical issue was identified with the JTT. While a fix was determined, manually testing to validate the solution was time-consuming.

With such high stakes, the SEC called on Eggplant's test automation to increase efficiencies, with the additional benefit of identifying and quickly resolving other unknown defects in the software.

Eggplant's intelligent, non-invasive technology enabled the SEC to reduce time spent manually testing and ensure safer, more reliable software for our joint warfighters.

Here's what the US Army said about its experience with Eggplant:

> Using Eggplant, the team was able to replicate the tests for different scenarios required. It could then execute the tests during off-hours to complete multiple test runs without a tester being present. Eggplant enabled the team to move from completing a 10-hour test event every three days to running a 24-hour test event every day.
>
> Ultimately, the JTT team ran approximately 90 hours of tests on off-hours. Using automation, it also uncovered additional problems inherent in the software, which enabled the developers to create solutions. It's now keeping the Eggplant scripts to use in future regression tests.

Learn more

Try Eggplant today to learn the following:

- how a two-system, model-based approach can automate any software written in any language on any device
- the importance of understanding test coverage and ways to improve on this to ultimately deliver better quality software
- the power and versatility of AI-driven, non-invasive test automation and its potential to transform your development and delivery of mission-critical software

Request a free trial account to start using Eggplant today.

US Navy image information

Cover image: ATLANTIC OCEAN (April 26, 2021) Rear Adm. Craig Clapperton, commander, Carrier Strike Group (CSG) 12, front, observes an air defense exercise in the Tactical Flag Command Center aboard USS Gerald R. Ford (CVN 78), April 26, 2021. (This image has been altered by blurring out computer screens for security purposes.) (U.S. Navy photo by Mass Communication Specialist 1st Class Jeff Troutman)

Image on page 10: ATLANTIC OCEAN (April 26, 2021) Lt. Cmdr. Josh Money, left, Carrier Strike Group (CSG) 12 Strike Officer, mans an air and missile defense coordination console during an air defense exercise in the Tactical Flag Command Center Aboard USS Gerald R. Ford (CVN 78), April 26, 2021. (This image has been altered by blurring out computer screens for security purposes.) (U.S. Navy photo by Mass Communication Specialist 1st Class Jeff Troutman)

Image on page 11: 190919-N-SE292-0007 SAN DIEGO (Sept. 19, 2019) Lt. j.g. Chastitie Polk, left, Lt. j.g. Cara Pastrana, Lt. j.g. Betty Yi, Lt. j.g. John Martin, Mo Okita, training systems lead for the Sea Combat Division at the Naval Surface and Mine Warfighting Development Center (SMWDC), Lt. j.g. Amber Glitz, and Chief Sonar Technician Travis Daniels, from the Center for Surface Combat Systems (CSCS), Detachment San Diego, work on computer simulations. SMWDC and CSCS host ASWO students for hands-on training inside of CSCS's Combined Integrated Air & Missile Defense/Anti-Submarine Warfare Trainer. (U.S. Navy photo by Clinton Beaird/Released)